SPI-VÄYLÄN TOTEUTUS FPGA-PIIRILLE

Lauri Similä

Ohjaaja: Jukka Lahti

ELEKTRONIIKAN JA TIETOLIIKENNETEKNIIKAN TUTKINTO-OHJELMA

2018

Similä L. (2018) SPI-väylän toteutus FPGA-piirille. Oulun yliopisto, Elektroniikan ja tietoliikennetekniikan tutkinto-ohjelma. Kandidaatintyö, 19 s

TIIVISTELMÄ

Tässä työssä toteutetaan SPI-väyläprotokolla SystemVerilog-kovonkuvauskielellä. Suunnittelun pohjana käytetään Motorolan SPI-väyläprotokollaa. Aluksi perehdytään väyläprotokollan teoriaan, minkä jälkeen luodaan SPI-väylän rekisterisiirtotason malli. Mallin toiminta varmennetaan simulointiohjelmalla, ja sille ajetaan FPGA-synteesi. Lopullista porttitason mallia verrataan rekisterisiirtotason malliin ja todetaan mallien yhtäpitävyys.

Avainsanat: SystemVerilog, RTL

Similä L. (2018) Implementation of SPI on FPGA board. University of Oulu, Degree Programme in Electronics and Communications Engineering, Bachelor's Thesis, 19 p.

ABSTRACT

In this project, a Serial Peripheral Interface is implemented with SystemVerilog hardware description language. The design is based on Motorola's SPI specification. At first, the theory of Serial Peripheral Interface is presented, and after that, a register-transfer level design is created. The logic behaviour of the design is verified by simulation. The design is synthesized for a specific FPGA board. In the end, the final gate level model is compared to the register-transfer level model.

Keywords: SystemVerilog, RTL

SISÄLLYSLUETTELO

ALKULAUSE

Tämä kandidaatintyö on nähnyt monta auringonlaskua. Monet yön pimeät tunnit ovat hioneet sen siihen loistoon, jolla se nyt valaisee lukijoidensa verkkokalvoja. Kaikki hyvä kestää aikansa, ja tämäkin työ kukoistaa vain hetkisen verran, kunnes se unohdetaan lukemattomien muiden bittien virtaan arkistojen syövereihin.

Haluan kiittää Kastarin ja Vanillan henkilökuntaa epäsuorasta, joskin elintärkeästä panoksestanne tähän työhön. Kiitos myös JJ:lle ja JBL Charge 3:lle. Teistä on myös ollut apua tällä, lähes ikuisuudelta tuntuneella matkalla.

Oulussa, marraskuussa 2018

Lauri Similä

LYHENTEIDEN JA MERKKIEN SELITYKSET

SPI	Serial Peripheral Interface, sarjamuotoinen oheislaiteliitäntä
SCK	Serial Clock, sarjakello
MOSI	Master Output, Slave Input, datasignaali isäntälaitteelta renkilaitteelle
MISO	Master Input, Slave Output, datasignaali renkilaitteelta isäntälaitteelle
SS	Slave Select, renkilaitteen valintasignaali
CPOL	Clock Polarity, kellosignaalin polariteetti
CPHA	Clock Phase, kellosignaalin vaihe
MSB	Most Significant Bit, merkitsevin bitti
LSB	Least Significant Bit, vähiten merkitsevä bitti
I/O	Input/Output, sisääntulo/ulostulo
LUT	Look-Up Table, hakutaulukko
FPGA	Field-Programmable Gate-Array, uudelleenohjelmoitava mikropiiri
RTL	Register-Transfer Level, rekisterisiirtotaso
ASM	Algorithmic State Machine, tilakaavio

1 JOHDANTO

SPI (Serial Peripheral Interface) on lyhyen kantaman väyläteknologia, joka on erilaisine variaatioineen laajasti käytössä monissa nykyisissä sulautetuissa järjestelmissä. Tässä työssä toteutetaan yksinkertainen SPI-väylä FPGA-piirille ja käytännön työ tehdään Digitaalitekniikka 2 -kurssin työympäristössä.

Työ koostuu teoria- ja toteutusosiosta. Teoriaosuudessa perehdytään SPI-väylän rakenteeseen ja toimintaan yleisellä tasolla. Teorian ja SPI-väylän spesifikaation pohjalta siirrytään toteutusosioon, jossa luodaan väylän RTL-malli, ja esitellään sen toiminta. RTL-mallista syntetisoidaan lopullinen porttitason malli, jonka oikeanlainen toiminta varmennetaan vertaamalla näitä kahta mallia toisiinsa. Vaikka SPI-väylä syntetisoidaankin yksittäiselle FPGA-piirille, on synteesi helppo ajaa melkein mille tahansa FPGA-piirille.

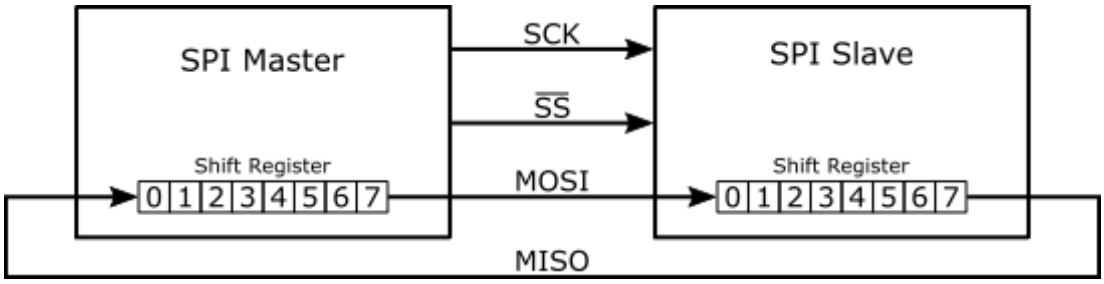
2 SPI-VÄYLÄ

2.1 Rakenne

SPI-väyläprotokolla on Motorolan kehittämä nelijohtiminen väyläliitäntä. Sitä käytetään yleisimmin lyhyen kantaman tiedonsiirtoon sulautetuissa järjestelmissä. Tyypillisiä sovelluskohteita ovat muun muassa SD-kortit, erilaiset sensorit ja nestekidenäytöt.

SPI-väylä koostuu yhdestä isäntälaitteesta ja yhdestä (tai useammasta) renkilaitteesta, jotka kommunikoivat keskenään kaksisuuntaisessa järjestelmässä (full duplex). Tiedonsiirto tapahtuu neljän signaalin avulla: kellosignaali SCK (Serial Clock), MOSI (Master Output, Slave Input), MISO (Master Input, Slave Output) ja SS (Slave Select). SPI-väylän rakenne on kuvattu kuvassa 1.

Isäntälaite generoi kellosignaalin SCK ja ohjailee renkilaitetta ennalta päätettyjen kellojaksojen ajan. Kellosignaali on valittava siten, että kellotaajuus sopii renkilaitteelle. [1]



Kuva 1. SPI-väylän rakenne.

2.2 Väylän toiminta

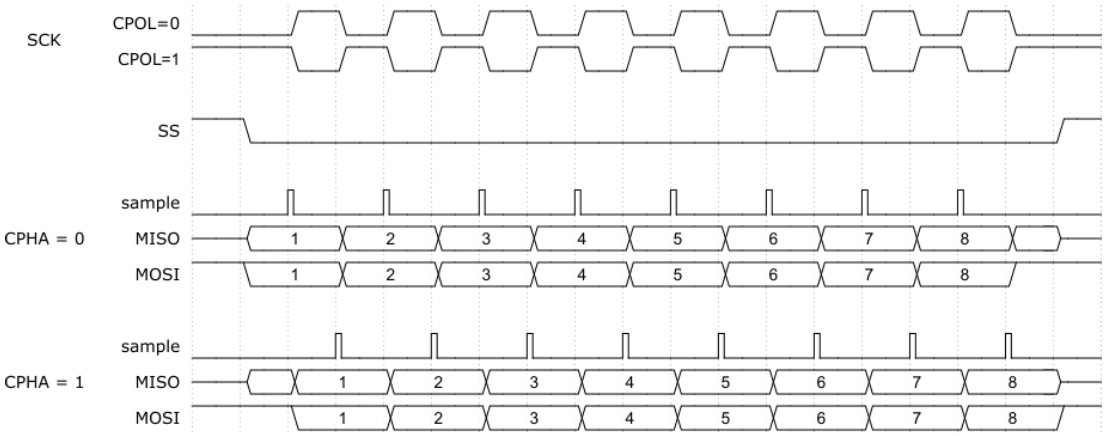
Tiedonsiirto alkaa kellosignaalin SCK määrittämisellä isäntälaitteessa, minkä jälkeen isäntälaite valitsee renkilaitteen asettamalla nolla-aktiivisen SS-signaalin tilaan 0. Isäntälaite lähettää bittejä renkilaitteelle MOSI-linjaa pitkin, ja renkilaite lähettää samaan aikaan dataa isäntälaitteelle (full duplex). Tiedonsiirron jälkeen isäntälaite asettaa SS-signaalin takaisin tilaan 1 ja lopettaa SCK-signaalin ajamisen.

Tyypillisesti sekä isäntä- että renkilaitteessa on siirtorekisteri, jossa bittejä siirretään kohti rekisterin MSB-paikkaa (Most Significant Bit). Samaan aikaan, kun isäntälaitteen siirtorekisterin sisältö siirretään bitti kerrallaan renkilaitteeseen merkitsevin bitti ensin, myös renkilaitteen rekisterin sisältö siirretään samalla tavalla isäntälaitteeseen. Kun tiedonsiirto on saatu päätökseen, isäntä ja renki ovat

vaihtaneet rekisteriensä sisällön keskenään. Mikäli tiedonsiirtoa halutaan jatkaa, ladataan rekistereihin uusi sisältö ja toistetaan tapahtumaketju. [1]

2.3 Kellosignaalin poratiteetti ja vaihe

Isäntälaitteen on määriteltävä kellosignaalin SCK polariteetti CPOL ja vaihe CPHA. Polariteetin ollessa 0 kellosignaali on tilassa 0, kun tiedonsiirtoa ei tapahdu renkilaitteen välillä. Vastaavasti, kun CPOL on 1, kellosignaali on tilassa 1. Kellosignaalin vaiheella tarkoitetaan sitä, millä kellonreunalla dataa luetaan rekistereihin. Kun CPHA=0, rekisterit ladataan parittomilla kellonreunoilla ja bittejä siirretään parillisilla kellonreunoilla. Vaiheen ollessa 1 rekisterit ladataan parillisilla kellonreunoilla ja bittejä siirretään parittomilla kellonreunoilla. Väylän toimintaa kuvaava ajoituskaavio on kuvassa 2. [1, 2]



Kuva 2. Ajoituskaavio

SPI-väylällä on erilaisia toimintatiloja eli polariteetin ja vaiheen kombinaatioita, jotka on numeroitu nollasta kolmoseen. Taulukossa 1 on kuvaus kustakin toimintatilasta.

Taulukko 1. SPI-väylän toimintatilat [1]

Toimintatila	CPOL	CPHA	Kuvaus
0	0	0	Rekisterit ladataan kellosignaalin nousevalla reunalla. Bittejä siirretään kellosignaalin laskevalla reunalla. Kellosignaali on tilassa 0, kun dataa ei lähetetä.
1	0	1	Rekisterit ladataan kellosignaalin laskevalla reunalla. Bittejä siirretään kellosignaalin nousevalla reunalla. Kellosignaali on tilassa 0, kun dataa ei lähetetä.
2	1	0	Rekisterit ladataan kellosignaalin laskevalla reunalla. Bittejä siirretään kellosignaalin nousevalla reunalla. Kellosignaali on tilassa 1, kun dataa ei lähetetä.
3	1	1	Rekisterit ladataan kellosignaalin nousevalla reunalla. Bittejä siirretään kellosignaalin laskevalla reunalla. Kellosignaali on tilassa 1, kun dataa ei lähetetä.

2.4 SPI-väylän hyvät ja huonot puolet

SPI-väylä on hyvin yleinen tiedonsiirtoväylä nykyisissä sulautetuissa järjestelmissä. Sen etuja ovat nopeus ja yksinkertaisuus. Rinnakkaismuotoisiin väyläratkaisuihin verrattuna SPI-väylä on edullisempi ja yksinkertaisempi, koska mikropiiriltä se vaatii vain neljä pinniä käyttöönsä. SPI-väylä ei tarvitse erillisiä synkronointipiirejä, koska renkilaitteet käyttävät isännän generoimaa sarjakelloa. SPI-väylällä ei ole rajoituksia bittimäärän eikä kellotaajuuden suhteen, mikä tekee siitä hyvin monikäyttöisen rakenteen.

SPI-väylästä puuttuu signaali datan vastaanoton merkiksi: renkilaite ei voi lähettää tietoa isännälle siitä, milloin on ottanut datan vastaan. Väyläprotokollasta puuttuu myös mekanismi virheellisen datan havaitsemiseen ja korjaamiseen. Siitä huolimatta, että SPI-väylä on niin yleisesti käytetty, sitä ei ole virallisesti standardoitu. [1]

3 SPI-VÄYLÄN TOTEUTUS KOVONKUVAUSKIELELLÄ

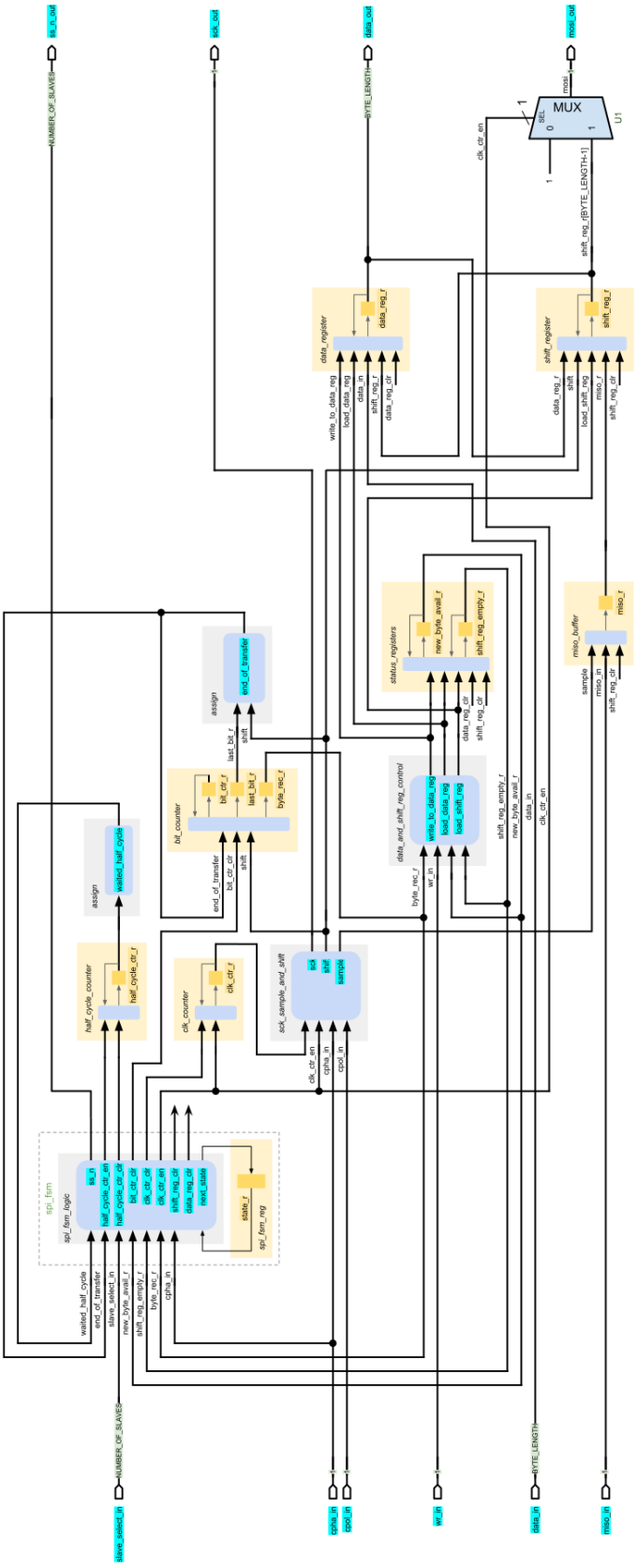
3.1 SystemVerilog

Tässä työssä suunniteltu SPI-väyläliityntä toteutettiin SystemVerilogilla. SystemVerilog on suosittu kovonkuvauskieli etenkin suunnittelun varmennuksessa, mutta sitä käytetään enenevissä määrin myös suunnittelukielenä. Se on Verilog-kielen uusin versio, jossa on piirteitä muun muassa VHDL- ja C-kielestä. SystemVerilog tuli viralliseksi IEEE-standardiksi vuonna 2005 ja korvasi edeltäjänsä kokonaan vuonna 2009.

SystemVerilogilla halutun piirin toiminta kuvataan rekisterisiirtotasolla tavallisesti always_comb- ja always_ff-prosesseilla sekä assign-lauseilla. Always_comb-prosessilla kuvataan pelkästään kombinaatiologiikkaa sisältävän lohkon toiminta. Mikäli lohkolle halutaan muistitoiminto, eli tallettaa bittejä rekistereihin, käytetään sekvenssilogiikalle tarkoitettua always_ff-prosessia. Assign-lauseita käytetään jatkuviin tai ehdollisiin signaalinasetuksiin. [3, 4]

3.2 Lohko- ja ASM-kaavio

Piirin sisään- ja ulostuloiksi nimettiin protokollan mukaisesti sarjakello sck_out, renkilaitteen valintasignaali ss_n_out sekä sarjamuotoisen datan sisääntulo miso_in ja ulostulo mosi_out. Ulkoisen kellosignaalin clk ja nollaussignaalin rst_n lisäksi sisääntuloiksi määriteltiin sarjakellon polariteetti cpol_in ja vaihe cpha_in sekä ulkoinen renkilaitteen valintasignaali slave_select_in. Datan kirjoitusta ja lukua varten tarvittiin datasignaalit data_in ja data_out sekä kirjoitussignaali wr_in. Toteutuksen lohkokaavio (kuvassa 3) suunniteltiin itse Motorolan SPI-protokollan [2] pohjalta.



Kuva 3. Lohkokaavio.

Laskurin clk_counter avulla generoidaan sarjakello sck sekä shift- ja sample-pulssit. Sallintasignaalin clk_ctr_en ollessa ykkönen clk_counter laskee clk-kellojaksoja parametrin SCK_PERIOD arvoon asti ja nollautuu. SCK_PERIOD kertoo sarjakellon jaksonpituuden mitattuna clk-kellojaksoina. Laskuria sallitaan vain DATA_TRANSFER-tilassa clk_ctr_en-signaalilla. Muulloin sitä nollataan clk_ctr_clr-signaalilla. Sarjakello sck dekoodataan laskurin arvosta, polariteetista cpol_in ja vaiheesta cpha_in. Puolivälissä sck-kellojaksoa generoidaan sample-pulssi, ja kellojakson lopuksi generoidaan shift-pulssi.

Datarekisteri data_register toimii sekä lähetettävän että vastaanotetun tavun säilytyspaikkana. Uusi tavu ladataan data_in:stä datarekisteriin, kun havaitaan write_to_data_reg-pulssi. Siirtorekisteristä ladataan renkilaitteelta vastaanotettu tavu datarekisteriin, jos havaitaan load_data-reg-pulssi. Datarekisterin kulloinenki sisältö näkyy suoraan data_out:ssa.

Siirtorekisterin shift_register tehtävä on hoitaa datansiirto isäntä- ja renkilaitteen välillä. Datarekisterissä lähetystä odottava tavu ladataan siirtorekisteriin load_shift_reg-pulssilla. Siirtorekisteriä sallitaan shift-pulssilla, jonka vaikutuksesta bittejä siirretään rekisterin sisällä kohti MSB-paikkaa ja miso_buffer:n sisältö ladataan LSB-paikalle. Renkilaitteelta tulevan datan puskuria miso_buffer:a sallitaan sample-pulssilla.

Lohkot status_registers ja data_and_shift_reg_control muodostavat kokonaisuuden, joka ohjailee sekä data- että siirtorekisteriä ja pitää kirjaa niiden tiloista. Datarekisterille lähetetään write_to_data_reg-pulssi, kun havaitaan wr_in-pulssi ja datarekisterissä ei ole tavua odottamassa lähetystä (new_byte_avail_r=0). Samalla asetetaan new_byte_avail_r-rekisterin arvo ykköseksi. Load_data_reg-pulssi lähetetään silloin, kun renkilaitteelta on vastaanotettu tavu ja new_byte_avail_r-rekisterin arvo on nolla. Tällöin myös shift_reg_empty_r-rekisterin arvo asetetaan ykköseksi. Jos datarekisterissä on tavu odottamassa lähetystä, niin siirtorekisterille lähetetään load_shift_reg-pulssi, mikäli siirtorekisteri on tyhjä (shift_reg_empty_r=1) tai sen jälkeen, kun renkilaitteelta on vastaanotettu tavu (byte_rec_r=1). Molemmissa tapauksissa rekisterien shift_reg_empty_r ja new_avail_r arvot asetetaan nolliksi.

Bit_counter-laskuri laskee vastaanotettuja bittejä BYTE_LENGTH:iin asti DATA_TRANSFER-tilassa. BYTE_LENGTH ilmaisee tavun koon bitteinä. Käytännössä laskuri laskee shift-pulsseja. Kun kaikki bitit on vastaanotettu, asetetaan byte_rec_r-rekisterin arvo ykköseksi.

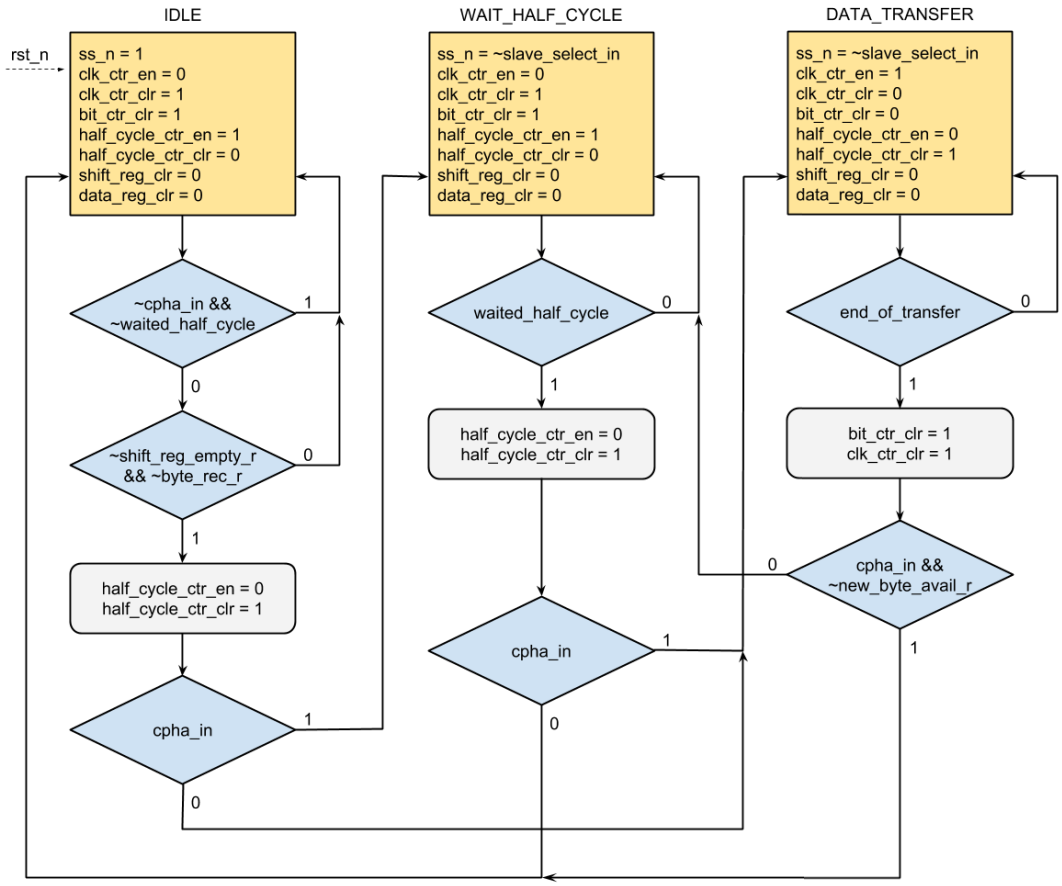
Toteutuksen kolmatta laskuria half_cycle_counter:a käytetään sekä IDLE- että WAIT_HALF_CYCLE-tilassa. Se laskee clk-kellojaksoja sck-kellojakson puoliväliin asti ja nollautuu vain half_cycle_ctr_clr-signaalin vaikutuksesta. Waited_half_cycle-signaali asetetaan ykköseksi yhtä clk-kellojaksoa ennen sck-kellojakson puoliväliä.

SPI-isäntälaitteella on kolme tilaa: IDLE, WAIT_HALF_CYCLE ja DATA_TRANSFER. IDLE-tilassa pysytään vähintään puolikkaan sck-kellojakson

verran, mikäli CPHA=0. Kun CPHA=1, voidaan siirtyä suoraan WAIT_HALF_CYCLE-tilaan. Renkilaitteen valintasignaali ss_n on ei-aktiivisessa tilassa (1). Seuraavaan tilaan siirrytään, kun siirtorekisteriin ladataan tavu lähetettäväksi datarekisteristä, eli kun shift_reg_empty_r menee nollaksi. Kun CPHA=0, seuraava tila on DATA_TRANSFER, ja kun CPHA=1, seuraava tila on WAIT_HALF_CYCLE. IDLE-tilasta siirryttäessä half_cycle_counter nollataan.

WAIT_HALF_CYCLE-tila on DATA_TRANSFER-tilaa ennen, jos CPHA=1, ja DATA_TRANSFER-tilan jälkeen, jos CPHA=0. Valintasignaalia ss_n ohjaa tässä tilassa slave_select_in, jossa tulee olla vain halutun renkilaitteen osalta bitti alhaalla koko lähetyksen ajan. Puolikkaan sck-kellojakson jälkeen half_cycle_counter nollataan ja siirrytään seuraavaan tilaan: IDLE-tilaan, jos CPHA=0, ja DATA_TRANSFER-tilaan, jos CPHA=1.

DATA_TRANSFER-tilassa ss_n on alhaalla halutun renkilaitteen osalta, clk_counter:a sallitaan clk_ctr_en-signaalilla ja bit_counter laskee vastaanotettuja bittejä. Viimeisen clk-kellojakson kohdalla, kun end_of_transfer-signaali käy ykkösenä, laskurit bit_counter ja clk_counter nollataan. IDLE-tilaan siirrytään vain, jos CPHA=1 ja uutta tavua ei ole odottamassa lähetystä datarekisterissä. Muuten siirrytään WAIT_HALF_CYCLE-tilaan. Kuvassa 4 on esitetty ASM-kaaviolla siirtymät eri tilojen välillä.



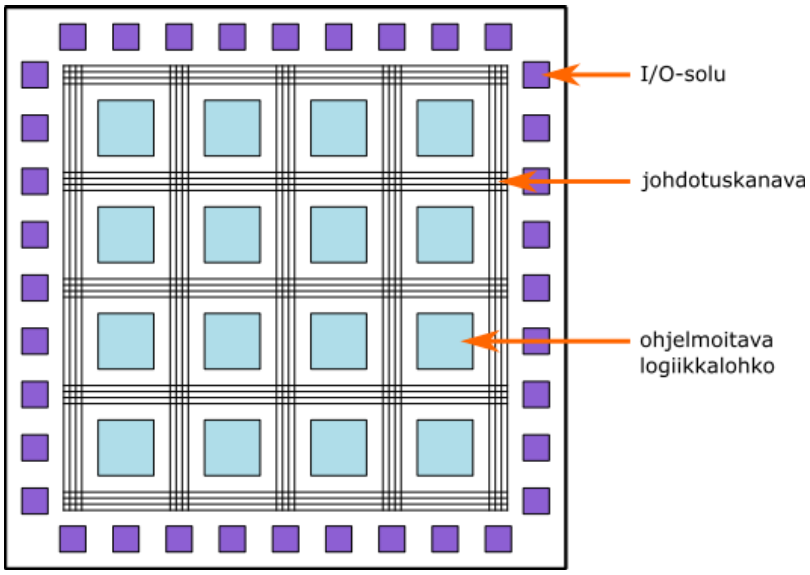
Kuva 4. ASM-kaavio.

3.3 FPGA-piiri

FPGA-piiri (field-programmable gate-array) on mikropiiri, jonka toiminta perustuu uudelleenkonfiguroitavaan logiikkamatriisiin. Matriisi koostuu ohjelmoitavista logiikkasoluista, I/O-soluista, muistilohkoista ja johdotuskanavista. Tyypillinen logiikkasolu pitää sisällään LUT:n, kokosummaimen ja D-kiikun. LUT:n sisään- ja ulostulot on kytketty johdotuskanaviin, joiden risteyskohdat ovat myös ohjelmoitavissa. I/O-solut voidaan konfiguroida sisään- tai ulostuloiksi halutun standardin mukaisesti. FPGA-piirillä on myös valmiina korkealaatuiset, valmiiksi kiikuille johdotetut kello- ja nollaussignaalit. Näiden lisäksi FPGA-piirillä on usein myös RAM-muisteja, DSP-lohkoja, erilaisia väyläliityntöjä, kuten I2C, SPI ja UART, sekä sulautettu mikroprosessori.

FPGA-piirien suosio perustuu niiden uudelleenkonfiguroitavuuteen. Suunnittelussa tapahtuneet virheet on helppo korjata ja ladata uusi konfigurointitiedosto piirille. FPGA-piirit sopivatkin hyvin prototyyppien tekemiseen tai esimerkiksi ASIC-piirien (Application Specific Integrated Circuit) suunnittelun tukemiseen. Ne eivät kuitenkaan sovi suurien tuotantomäärien valmistamiseen tai

energiatehokkaisiin sovelluskohteisiin, sillä suurin osa niiden transistoreista kuluu ohjelmoitavuuden mahdollistamiseen. [5, 6]



Kuva 5. FPGA-piirin rakenne

3.4 Työn kulku

Työ toteutettiin Digitaalitekniikka 2 -kurssin työympäristössä. SPI-väylän RTL-mallille luotiin testipenkki, jossa piirin sisääntuloihin syötettiin dataa ja tarkkailtiin sekä piirin ulostuloja että sen sisäisiä signaaleja. Jokaiselle sarjakellon polariteetin ja vaiheen kombinaatiolle ajettiin samat testit. Malli simuloitiin Mentor Graphicsin QuestaSim-ohjelmalla. Simulaattorilla tarkasteltiin signaalien aaltomuotoja sekä simulaattorin luomaa piirikaaviota. Aaltomuotoja verrattiin Motorolan SPI-väyläprotokollan spesifikaatioon [2], ja tarvittaessa suunnittelussa tulleita virheitä korjattiin.

Kun simulointitulokset vastasivat spesifikaation kuvausta, ajettiin mallille FPGA-synteesi Alteran Quartus II -ohjelmalla. Kohdepiiriksi valittiin Alteran yksi halvimmista FPGA-piireistä, Cyclone V E -tuoteperheen jäsen 5CEFA2F23C8. FPGA-synteesissä rekisterisiirtotason mallista luotiin FPGA-piirivalmistajan komponenttikirjastojen avulla optimoitu porttitason malli. Tälle porttitason mallille ajettiin ajoitusanalyysi, tarkasteltiin mallin vaatimaa alaa FPGA-piiriltä ja varmistettiin mallin looginen toiminta tarkastelemalla signaalien aaltomuotoja QuestaSim-ohjelmalla. Tulosta verrattiin RTL-simulointituloksiin. Työssä ei käytetty fyysistä FPGA-piiriä.

3.5 Työn tulokset

Työssä onnistuttiin luomaan toimiva SPI-väyläprotokollan toteuttava malli valitulle FPGA-piirille. Porttitason simuloinnissa signaalien aaltomuotojen perusteella pääteltiin, että mallin toiminta vastasi RTL-tason toimintaa. Simulaattori ei havainnut ajoitusrikkomuksia, eikä muitakaan virheitä.

RTL-lähdekoodin perusteella rekistereitä laskettiin olevan 32 kappaletta. FPGA-synteesin jälkeen valmis porttitason malli tarvitsi käyttöönsä 30 logiikkalohkoa, 32 I/O-pinniä ja 40 rekisteriä. Varsinaisia RTL-koodia vastaavia rekistereitä syntetisoitui yksi enemmän kuin mitä RTL-koodin perusteella laskettiin. Tämä johtui siitä, että synteesiohjelma loi jokaiselle mallin tilalle (IDLE, WAIT_HALF_CYCLE, DATA_TRANSFER) oman rekisterinsä. RTL-koodissa tilojen koodaamiseen varattiin kaksi bittiä (rekisteriä). Muita resursseja porttitason malli ei tarvinnut piiriltä.

4 YHTEENVETO

Tässä kandidaatintyössä toteutettiin SPI-väyläprotokollan mukainen malli FPGA-piirille. Suunnittelukielenä oli SystemVerilog-kovonkuvauskieli, ja käytännön työ tehtiin Digitaalitekniikka 2 -kurssin työympäristössä. Simulointi- ja synteesityökaluina käytettiin Mentor Graphicsin QuestaSim ja Alteran Quartus II -ohjelmia.

Teoriaosuudessa esiteltiin SPI-väylän rakenne ja toiminta yleisellä tasolla. Toteutusosuudessa esiteltiin käytettävä kovonkuvauskieli. Lohko- ja ASM-kaavioiden avulla selitettiin mallin yksityiskohtainen toiminta RTL-tasolla. Työssä käytiin lyhyesti läpi myös FPGA-piirien toiminta ja suunnittelun eri vaiheet RTL-mallin luomisesta aina valmiin porttitason mallin syntetisoimiseen. Lopuksi esiteltiin työn tulokset.

Valmis porttitason malli vaatii FPGA-piiriltä 30 logiikkalohkoa, 32 I/O-pinniä ja 40 rekisteriä (D-kiikkua). Tämä resurssimäärä on sen verran pieni, että se mahtuu melkein mille tahansa FPGA-piirille.

5 LÄHTEET

[1] Wikipedia (luettu 2.11.2018) Serial Peripheral Interface. URL: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface

[2] Motorola (luettu 5.11.2018) SPI Block Guide V03.06. URL: https://opencores.org/usercontent/doc/1499360489

[3] Wikipedia (luettu 22.11.2018) SystemVerilog. URL: https://en.wikipedia.org/wiki/SystemVerilog

[4] What is SystemVerilog? (luettu 22.11.2018) URL: https://www.doulos.com/knowhow/sysverilog/whatissv/

[5] Digitaalitekniikka 2 (luettu 25.11.2018) URL: https://optima.oulu.fi/learning/id76/bin/doc_show?id=349677

[6] Digitaalitekniikka 3 (luettu 25.11.2018) URL: https://optima.oulu.fi/learning/id76/bin/doc_show?id=316436